# PHP Leaves Management Service

A full-stack vacation/leave management system built with PHP backend and React frontend.

Contents

[PHP Leaves Management Service 1](#_Toc212416539)

[Features 2](#_Toc212416540)

[Technology Stack 2](#_Toc212416541)

[Backend 2](#_Toc212416542)

[Frontend 2](#_Toc212416543)

[Installation Guide 3](#_Toc212416544)

[Prerequisites 3](#_Toc212416545)

[Step 1: Database Setup 3](#_Toc212416546)

[Step 2: Backend Setup 4](#_Toc212416547)

[Step 3: Frontend Setup 4](#_Toc212416548)

[Running the Application 5](#_Toc212416549)

[Start Backend Server 5](#_Toc212416550)

[Start Frontend Development Server 5](#_Toc212416551)

[Access the Application 5](#_Toc212416552)

[Testing 6](#_Toc212416553)

[What Are Unit Tests? 6](#_Toc212416554)

[Running Unit Tests 6](#_Toc212416555)

[Test Suites 7](#_Toc212416556)

[Running Specific Tests 7](#_Toc212416557)

[When to Run Tests 7](#_Toc212416558)

[Test Database 8](#_Toc212416559)

[User Manual 8](#_Toc212416560)

[Login Credentials 8](#_Toc212416561)

[Manager Features 8](#_Toc212416562)

[Employee Features 9](#_Toc212416563)

[Common Tasks 9](#_Toc212416564)

[Logout 10](#_Toc212416565)

[Project Structure 10](#_Toc212416566)

[Troubleshooting 11](#_Toc212416567)

[Database Connection Issues 11](#_Toc212416568)

[CORS Errors 11](#_Toc212416569)

[Port Already in Use 11](#_Toc212416570)

[Login Not Working 12](#_Toc212416571)

[PHPUnit Not Found 12](#_Toc212416572)

[React White Screen 12](#_Toc212416573)

[Cannot Create User 12](#_Toc212416574)

## Features

* **User Authentication** - Secure JWT-based login system
* **Role-Based Access Control** - Manager and Employee roles
* **Vacation Request Management** - Create, view, and track vacation requests
* **Manager Approval Workflow** - Approve or reject vacation requests
* **User Management** - Create and manage employee accounts
* **Work Schedule Management** - Track employee work schedules
* **Audit Logging** - Complete activity tracking

## Technology Stack

### Backend

* PHP 8.2+
* MySQL 5.7+
* PHPUnit for testing
* JWT authentication

### Frontend

* React 18
* React Router 6
* Axios for API calls
* Vite for development

## Installation Guide

### Prerequisites

Before you begin, ensure you have the following installed:

1. **XAMPP** (includes PHP and MySQL)
   * Download from: https://www.apachefriends.org/
   * Version: PHP 8.0 or higher
2. **Composer** (PHP package manager)
   * Download from: https://getcomposer.org/
   * Used for installing PHPUnit and dependencies
3. **Node.js and npm** (JavaScript runtime and package manager)
   * Download from: https://nodejs.org/
   * Version: Node 16+ recommended

### Step 1: Database Setup

#### 1.1 Start MySQL Server

1. Open **XAMPP Control Panel**
2. Click **Start** next to **MySQL**
3. Wait for the status to turn green

#### 1.2 Import Database Schema

1. Click on the **“Import”** tab
2. Click **“Choose File”**
3. Navigate to: phpLeavesService/installation/create-database-query.sql
4. Click **“Go”** at the bottom
5. Wait for success message: “Import has been successfully finished”

#### 1.3 Import Dummy Data

1. With leaves\_management database still selected
2. Click **“Import”** tab again
3. Choose file: phpLeavesService/installation/dummy-data.sql
4. Click **“Go”**
5. Success! You now have test users:
   * **Manager**: username alice, password password123
   * **Employee**: username bob, password password123

#### 1.5 Create Test Database (for unit tests)

1. Click on the **“Import”** tab
2. Click **“Choose File”**
3. Navigate to: phpLeavesService/installation/create-test-database-query.sql
4. Click **“Go”** at the bottom
5. Wait for success message: “Import has been successfully finished”

**Database setup complete!**

### Step 2: Backend Setup

#### 2.1 Configure Database Connection

1. Open file: backend/src/config/config.php
2. Verify the database settings:

'database' => [  
 'host' => 'localhost',  
 'db\_name' => 'leaves\_management',  
 'username' => 'root',  
 'password' => '', // Default XAMPP password is empty  
 'charset' => 'utf8mb4'  
]

1. If your MySQL has a password, update the password field

#### 2.2 Install PHP Dependencies

1. Open **Command Prompt** or **PowerShell**
2. Navigate to the backend folder:

* cd C:\path\to\phpLeavesService\backend

1. Install Composer dependencies:

* composer install

1. Wait for installation to complete
2. Verify PHPUnit is installed:

* php vendor/bin/phpunit --version
* Should show: PHPUnit 9.6.29

**Backend setup complete!**

### Step 3: Frontend Setup

#### 3.1 Install Node.js Dependencies

1. Open **Command Prompt** or **PowerShell**
2. Navigate to the frontend folder:

* cd C:\path\to\phpLeavesService\frontend

1. Install npm packages:

* npm install

1. Wait 1-2 minutes for installation (downloads ~200MB of packages)

**Frontend setup complete!**

## Running the Application

### Start Backend Server

1. Open **PowerShell** or **Command Prompt**
2. Navigate to project root:

* cd C:\path\to\phpLeavesService

1. Start PHP development server:

* php -S localhost:8080 -t backend/public backend/public/index.php

1. You should see:

* [Sun Oct 26 20:00:00 2025] PHP 8.2.12 Development Server (http://localhost:8080) started

1. **Keep this terminal open** - closing it stops the server

### Start Frontend Development Server

1. Open a **NEW** PowerShell or Command Prompt window
2. Navigate to frontend folder:

* cd C:\path\to\phpLeavesService\frontend

1. Start Vite development server:

* npm run dev

1. You should see:

* VITE v5.x.x ready in 500 ms  
  ➜ Local: http://localhost:3000/

1. **Keep this terminal open** as well

### Access the Application

1. Open your browser
2. Go to: **http://localhost:3000**
3. You should see the login page

**Application is now running!**

## Testing

### What Are Unit Tests?

Unit tests are automated tests that verify your code works correctly. They: - Catch bugs before users find them - Ensure new changes don’t break existing features - Document how code should behave - Give confidence when refactoring

### Running Unit Tests

#### Basic Test Run

1. Open **PowerShell** or **Command Prompt**
2. Navigate to backend folder:

* cd C:\path\to\phpLeavesService\backend

1. Run all tests:

* php vendor/bin/phpunit

#### Understanding Test Output

**Successful Test Run:**

PHPUnit 9.6.29 by Sebastian Bergmann and contributors.  
  
............. 13 / 13 (100%)  
  
Time: 00:01.644, Memory: 6.00 MB  
  
OK (13 tests, 30 assertions)

* **13 dots (.)** = 13 tests passed
* **OK** = All tests successful
* **13 tests** = Number of test methods executed
* **30 assertions** = Individual checks performed

**Failed Test Run:**

F............ 13 / 13 (100%)  
  
FAILURES!  
Tests: 13, Assertions: 29, Failures: 1.  
  
1) Tests\Unit\ValidatorTest::testEmailValidation  
Failed asserting that false is true.  
/path/to/ValidatorTest.php:42

* **F** = Failed test
* Shows which test failed and why
* Shows line number to fix

### Test Suites

Your application has **3 test suites**:

#### 1. Validator Tests (7 tests)

Location: tests/Unit/ValidatorTest.php

Tests input validation: - Email validation - Required field validation - String length validation - Numeric validation - Date validation

#### 2. Response Tests (2 tests)

Location: tests/Unit/ResponseTest.php

Tests API response formatting: - JSON response structure - Error response format

#### 3. Authentication Tests (4 tests)

Location: tests/Integration/AuthControllerTest.php

Tests login functionality: - Successful login - Invalid credentials - Missing fields - JWT token generation

### Running Specific Tests

**Run only Unit tests:**

php vendor/bin/phpunit --testsuite "Unit Tests"

**Run only Integration tests:**

php vendor/bin/phpunit --testsuite "Integration Tests"

**Run a specific test file:**

php vendor/bin/phpunit tests/Unit/ValidatorTest.php

### When to Run Tests

* Before committing code changes
* After fixing a bug
* After adding new features
* Before deploying to production

### Test Database

Tests use a **separate database**: leaves\_management\_test

This ensures: - Tests don’t corrupt production data - Tests can safely delete/modify records - Tests are isolated and repeatable

## User Manual

### Login Credentials

**Default Test Accounts:**

| Username | Password | Role | Permissions |
| --- | --- | --- | --- |
| alice | password123 | Manager | View/manage all users and requests |
| bob | password123 | Employee | Create and view own requests |

### Manager Features

#### 1. Dashboard

* View pending vacation requests requiring approval
* See total users and requests statistics
* Quick access to approval actions

#### 2. User Management

**Access:** Click “Users” in navigation menu

**View Users:** - See list of all employees - View user details (name, email, role, schedule) - Search and filter users

**Create User:** 1. Click **“Add User”** button 2. Fill in the form: - **First Name** (required) - **Last Name** (required) - **Email** (required, valid format) - **Username** (required, unique) - **Password** (required, min 6 characters) - **Role** (Manager or Employee) - **Work Schedule** (Full-time/Part-time/Contractor) - **Vacation Days** (annual allowance) 3. Click **“Create User”**

**Edit User:** 1. Click **“Edit”** button next to user 2. Modify fields (password is optional) 3. Click **“Update User”**

**Delete User:** 1. Click **“Delete”** button next to user 2. Confirm deletion 3. User is permanently removed

#### 3. Vacation Approvals

**Access:** Click “Vacation Approvals” in navigation menu

**View Pending Requests:** - See all vacation requests awaiting approval - View employee name, dates, duration, reason

**Approve Request:** 1. Find the request 2. Click **“Approve”** button 3. Request status changes to “Approved” 4. Employee is notified

**Reject Request:** 1. Find the request 2. Click **“Reject”** button 3. Request status changes to “Rejected” 4. Employee is notified

**Filter Requests:** - **All** - Show all requests - **Pending** - Show only awaiting approval - **Approved** - Show approved requests - **Rejected** - Show rejected requests

### Employee Features

#### 1. My Requests

**Access:** Click “My Requests” in navigation menu

**View Your Requests:** - See all your vacation requests - Check status: Pending, Approved, or Rejected - View request dates and duration - See remaining vacation days

#### 2. Request Vacation

**Access:** Click “Request Vacation” in navigation menu

**Submit New Request:** 1. Click **“Request Vacation”** button 2. Fill in the form: - **Start Date** (date picker) - **End Date** (date picker, must be after start date) - **Reason** (optional, explain purpose) 3. Review calculated **days requested** 4. Check your **remaining vacation days** 5. Click **“Submit Request”** 6. Wait for manager approval

**Request Rules:** - End date must be after start date - Cannot request more days than available - Cannot request past dates - Weekends may or may not count (depends on schedule)

#### 3. Request Status

**Status Types:** - **Pending** - Waiting for manager approval - **Approved** - Request accepted, vacation confirmed - **Rejected** - Request denied by manager

### Common Tasks

#### Change Your Password

1. Log in to the application
2. Contact your manager to reset password
3. (Password change feature coming soon)

#### Check Vacation Balance

1. Go to **“Request Vacation”** page
2. See **“Remaining Days”** at the top
3. Shows current available vacation days

#### View Request History

1. Go to **“My Requests”** (employees) or **“Vacation Approvals”** (managers)
2. Use filter dropdown to see past requests
3. Review dates and statuses

### Logout

* Click **“Logout”** button in top-right corner
* You’ll be returned to login page
* Session is cleared securely

## Project Structure

phpLeavesService/  
├── backend/  
│ ├── public/  
│ │ ├── index.php # Main router  
│ │ └── api/  
│ │ ├── auth.php # Login/logout endpoints  
│ │ ├── users.php # User CRUD operations  
│ │ ├── requests.php # Vacation requests  
│ │ ├── schedules.php # Work schedules  
│ │ └── lookups.php # Dropdown data (roles, statuses)  
│ ├── src/  
│ │ ├── config/  
│ │ │ ├── config.php # Application configuration  
│ │ │ └── database.php # Database connection  
│ │ ├── controllers/ # Business logic  
│ │ ├── models/ # Database models  
│ │ └── utils/ # Helper classes (Auth, Validator, Response)  
│ ├── tests/  
│ │ ├── bootstrap.php # Test initialization  
│ │ ├── Unit/ # Unit tests  
│ │ └── Integration/ # Integration tests  
│ ├── composer.json # PHP dependencies  
│ └── phpunit.xml # PHPUnit configuration  
├── frontend/  
│ ├── src/  
│ │ ├── api/  
│ │ │ └── axios.js # API client configuration  
│ │ ├── components/ # Reusable React components  
│ │ ├── context/  
│ │ │ └── AuthContext.jsx # Authentication state  
│ │ ├── pages/  
│ │ │ ├── Manager/ # Manager-only pages  
│ │ │ ├── Employee/ # Employee-only pages  
│ │ │ └── SignIn.jsx # Login page  
│ │ ├── App.jsx # Main app component  
│ │ └── main.jsx # React entry point  
│ ├── package.json # Node.js dependencies  
│ └── vite.config.js # Vite configuration  
└── installation/  
 ├── create-query.sql # Production database schema  
 ├── create-test-query.sql # Test database schema  
 └── dummy-data.sql # Sample data

## Troubleshooting

### Database Connection Issues

**Problem:** “Database connection failed”

**Solutions:** 1. Ensure MySQL is running in XAMPP Control Panel 2. Check credentials in backend/src/config/config.php 3. Verify database name is correct: leaves\_management 4. Default XAMPP password is empty string

### CORS Errors

**Problem:** “Access to XMLHttpRequest has been blocked by CORS policy”

**Solutions:** 1. Ensure backend server is running on port 8080 2. Check frontend/src/api/axios.js has correct baseURL: http://localhost:8080 3. Verify all API files have CORS headers applied

### Port Already in Use

**Problem:** “Address already in use” when starting server

**Solution for Port 8080:**

# Find process using port 8080  
netstat -ano | findstr :8080  
  
# Kill the process (replace PID with actual process ID)  
taskkill /PID <PID> /F

**Solution for Port 3000:**

# Use a different port  
npm run dev -- --port 3001

### Login Not Working

**Problem:** Login button does nothing or shows errors

**Checklist:** 1. Backend server running on port 8080 2. Frontend server running on port 3000 3. Database leaves\_management exists and has data 4. Using correct credentials (alice/password123 or bob/password123) 5. Check browser console (F12) for error messages

### PHPUnit Not Found

**Problem:** “phpunit: command not found”

**Solution:** 1. Run from backend folder: cd backend 2. Use full path: php vendor/bin/phpunit 3. Or reinstall: composer install

### React White Screen

**Problem:** Frontend shows blank white page

**Solutions:** 1. Check browser console (F12) for errors 2. Clear browser cache (Ctrl+Shift+Delete) 3. Clear localStorage: - Open browser console (F12) - Type: localStorage.clear() - Refresh page 4. Verify frontend server is running without errors

### Cannot Create User

**Problem:** Creating User errors

**Common Causes:** 1. Username already exists (must be unique) 2. Email already exists (must be unique) 3. Invalid email format 4. Password too short (minimum 6 characters) 5. Missing required fields